Cluster-Based Repositories and Analysis

Technical Report NIM-2004-001

Contract: DAAH01-03-C-R219
Final Report
Reporting Period:
27 May 2003— end of project

Prepared by:
Alok Choudhary, Ph.D.
Nimkathana Corporation
1807 W. Winnemac, Unit A
Chicago, IL 60640
773.515.2562
choudhary@nimkathana.com

1 Overall Status

This is the final report for the DARPA SBIR project "Cluster-Based Repositories and Analysis". Tasks described in our original proposal have been completed. Drs. Alok Choudhary and George Thiruvathukal met with Lt Col Dr. Doug Dyer on July 23, 2003 at DARPA to present the progress on this project and obtain feedback.

2 Task Description and Status

2.1 Task: Scalable Commodity-based Cluster

We designed and configured a Linux-based cluster with 1 terabyte of storage, parallelism in storage, and several computing nodes for parallel computing using commodity components as described in the proposal.

Figure 1 shows the configuration of the cluster. The cluster has the following characteristics and specification. Nimkathana bought all commodity components and built the cluster using in-house experience.

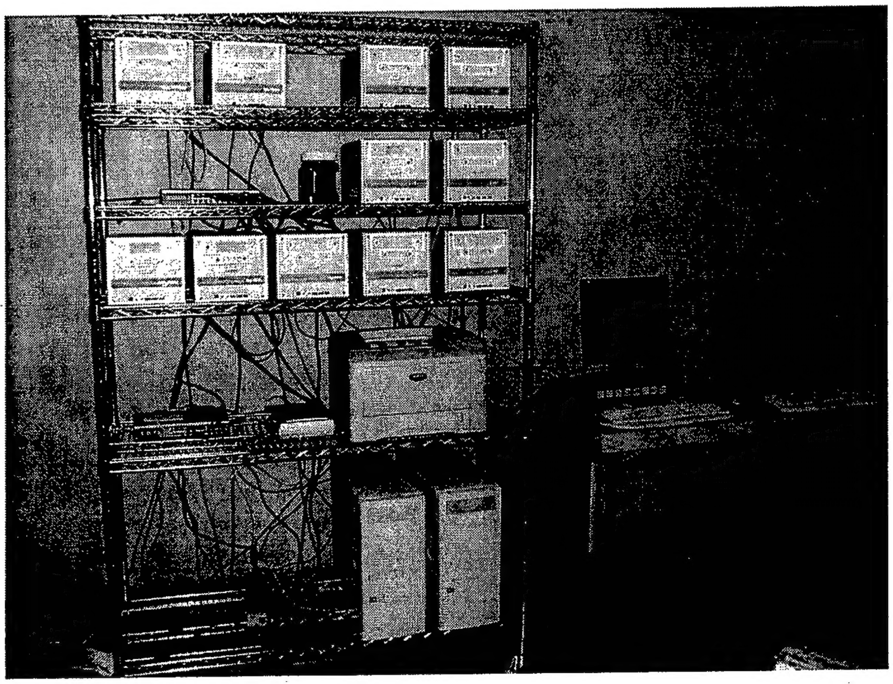

Figure 1: Figure of the Configured Cluster

- 8-node, 1 Terabyte (TB) Cluster
- 120GB (1 TB) Local Disk per node
- AMD Athlon XP 2400+ (2.26 GHz)
- Switched Ethernet 100 Mbps (for now)

- 1.25 TB RAID Server
- ATA -> SCSI RAID Controller
- Hot-Swappable Drives
- 1 TB Usable Filesystem Storage
- Standard RedHat 9.0
- Private Network
- DHCP, NIS, NFS (for shared software and authentication); OpenLDAP later.
- MPICH Distribution for communication.

2.2 Task: Software Architecture

Figure 2 illustrates the overall software architecture, which comprises layers for parallel file systems; parallel I/O and a communication runtime system; software infrastructure to enable basic primitives and functions found in parallel data mining algorithms with well defined interfaces. This architecture was presented and explained to Lt Col Dr. Doug Dyer on July 23, 2003. As part of phase I, components of this architecture were to be prototyped. We have completed the design of the architecture.

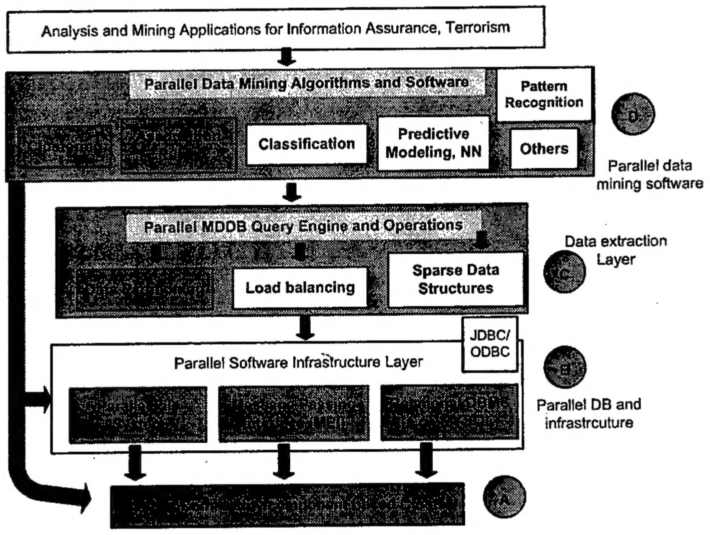
Analysis and Mining Applications for Information Assurance, Terrorism
Parallel Data Mining Algorithms and Software
Pattern Recognition
Classification
Predictive Modeling, NN
Others
D
Parallel data mining software
Parallel MDDB Query Engine and Operations
Load balancing
Sparse Data Structures
Data extraction Layer
C
JDBC/ ODBC
Parallel Software Infrastructure Layer
B
Parallel DB and infrastrcuture
A

Figure 2: Overall Software Architecture for Scalable Data Analysis and Mining on Commodity Clusters

2.3 Task: Parallel Software Infrastructure

We proposed to port, implement and evaluate public domain software for communication and I/O within the cluster based on the the Message Passing Interface (MPI).

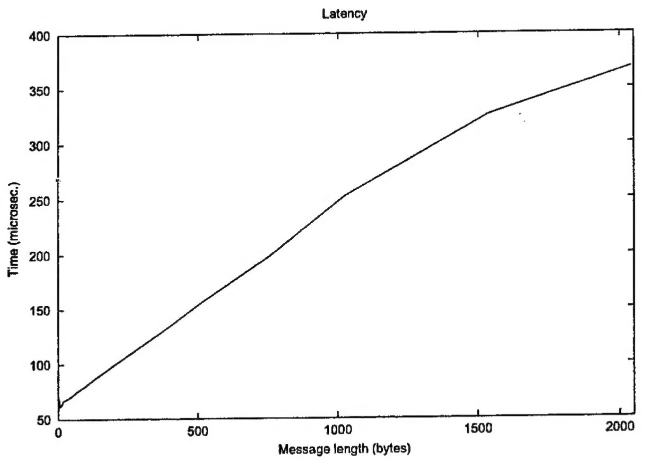

Figure 3: Communication latency on the cluster. This communicaiton latency is the latency seen by the software that uses the communication primitives. It is obtained using MPI functions.

2.3.1 Subtask: Parallel Communication Software

We ported and installed the Message Passing Interface (MPI) standard on the cluster testbed to enable high-performance communication between cluster nodes (See `http://www.mpi-forum.org/`). We installed MPICH-1.2.5 on the cluster. MPICH (`http://www.mcs.anl.gov/mpi/mpich`) is a popular, freely available implementation of the Message Passing Interface (MPI) Standard. We ran some basic performance tests to measure the performance of MPICH on the cluster. We used an MPI version of the Netpipe benchmark to measure the latency and bandwidth. For communication between two nodes of the cluster, the MPI latency was measured to be about 60 μs for a 0-byte message as shown in Figure 7. The bandwidth flattened out to about 85 Mb/s for messages of size 32 Kbytes and higher (Figure 8). The Fast Ethernet network on the cluster has a maximum bandwidth of 100 Mb/s, and we are able to get 85% of that through MPI. We also ran some tests to measure the performance of collective communication algorithms in MPI, such as broadcast. For this purpose we used the `mpptest` benchmark that is available as part of MPICH. Figure 5 shows the performance of `MPI_Bcast` on 3–8 nodes of the cluster for different message sizes. The graphs illustrate the power-of-two nature of the binary tree algorithm used to implement `MPI_Bcast` for these message sizes. The performance on 3 and 4 nodes is about the same because they fall within the same power of two (4). Similarly, the performance for 5, 6, 7, and 8 nodes is about the same because they fall within the same power of two (8).

We also tested the performance instrumentation and visualization facilities available in MPICH, namely the MPE logging library and *Jumpshot* performance visualizer. Figure 6 shows the Jumpshot view for a program that calculates the value of *pi* in parallel. The figure shows time on the X-axis and the processes (8) on the Y-axis. The horizontal bars show the time taken for computation and MPI communication operations used in this program, namely, `MPI_Bcast`, `MPI_Reduce` (global sum), and `MPI_Barrier`.

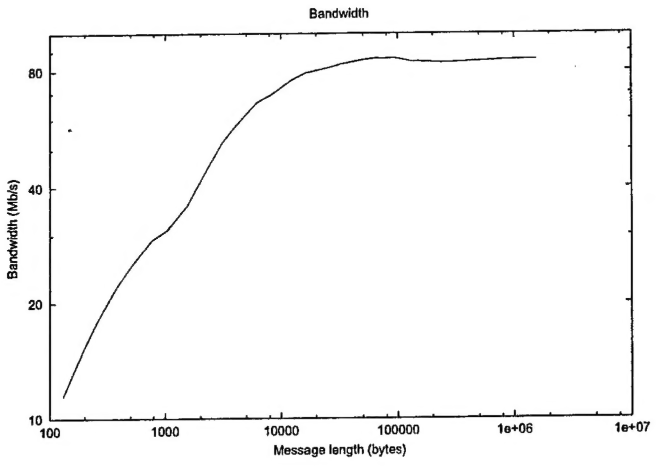

Figure 4: Communication bandwidth on the cluster. This communicaiton bandwidth is the bandwidth seen by the software that uses the communication primitives. It is obtained using MPI functions.

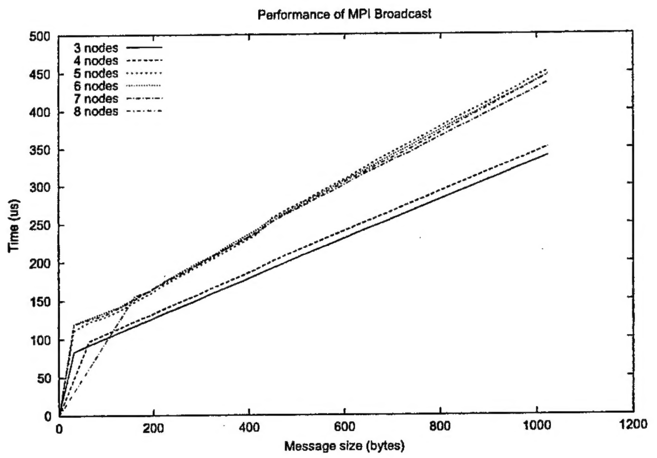

Figure 5: Performance of Broadcast function on the cluster.

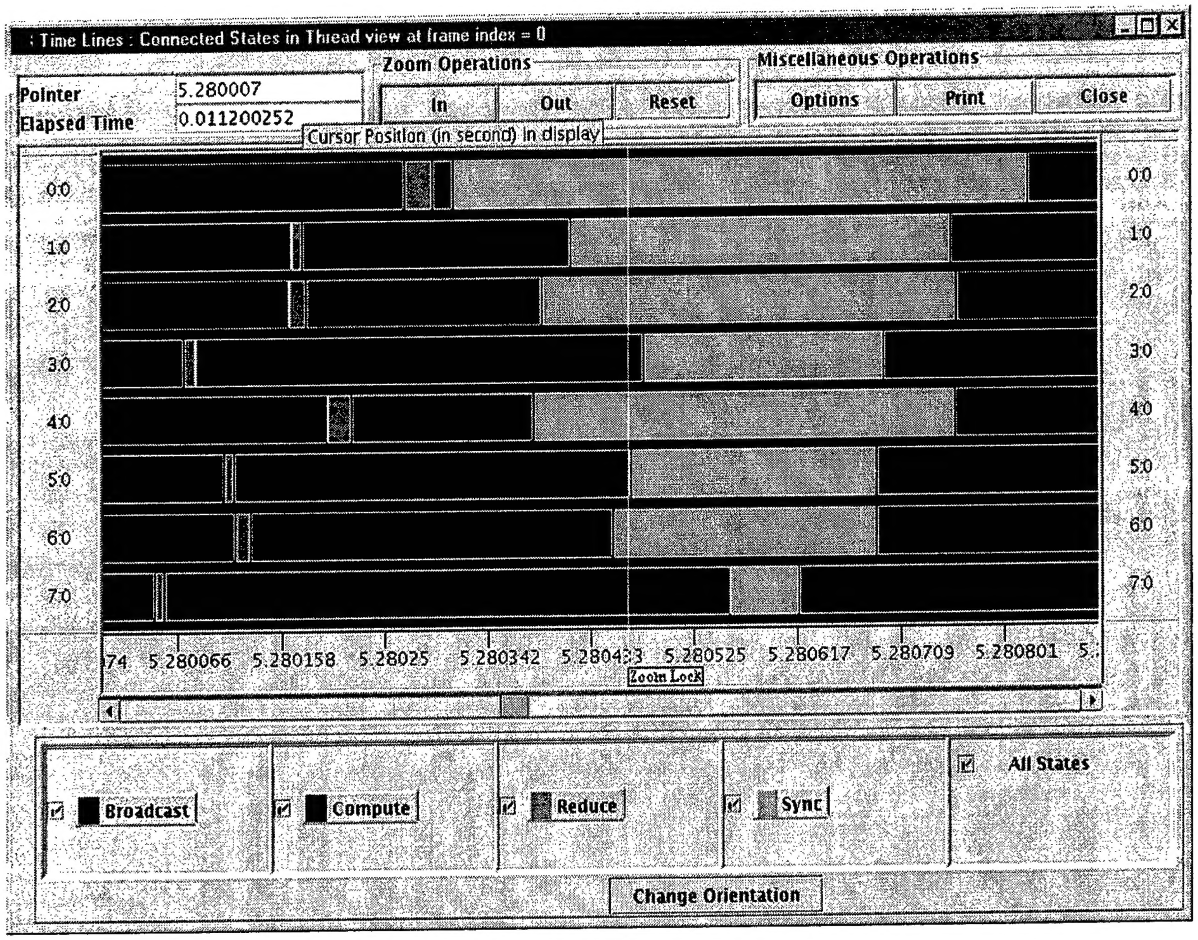

Figure 6: Performance Instrumentation and Visualization of various components of communication on the cluster.

2.3.2 MPICH-2

We installed MPICH-2 on the cluster and compared the performance with MPICH-1. MPICH-2 is an all-new MPI implementation being developed at Argonne National Laboratory (`http://www.mcs.anl.gov/mpi/mpich2`). Among its new features is a much faster implementation for TCP, faster process startup, new collective communication algorithms, better derived datatype support, and an almost-complete implementatoin of MPI-2 functionality, including one-sided communication, dynamic process management, and parallel I/O. MPICH-2 is implemented directly on top of TCP (sockets), unlike MPICH-1 which was layered on top of p4. p4 is a complete communication library in its own right, and it introduces an extra memory copy on all messages. We compared the performance of MPICH-2 (version 0.96 beta) and MPICH-1 (version 1.2.5) on our cluster. The results are presented below.

We first ran some basic performance tests to measure the latency and bandwith using the Netpipe benchmark. Figure 7 shows the latency for communication between two nodes of the cluster. For MPICH-1.2.5, we measured the latency to be 60 μs for a 0-byte message, whereas with MPICH-2 it was 46 μs. This 23% improvement in latency is primarily due to the elimination of p4 as a communication layer in MPICH-2. The bandwidth results are shown in Figure 8. The bandwidth is limited by the bandwidth of fast ethernet (100 Mb/s). MPICH-2 saturates at about 90 Mb/s, whereas MPICH-1 gives about 85 Mb/s. The higher bandwidth is because of fewer memory copies.

We also ran some tests to compare the performance of collective communication. Figures 9– 11 show the performance of `MPI_Reduce` global sum operations for arrays of size 512K to 2M integers. MPICH-1 uses the standard binary tree algorithm to do a reduction, whereas MPICH-2 for large messages uses a more sophisticated algorithm that minimizes bandwidth usage. It implements the reduce as a reduce-scatter followed by a gather. As a result, the performance is better than the reduce in MPICH-1, as shown in Figures 9– 11.

We also measured the performance of broadcast in MPICH-1 and MPICH-2 (see Figures 9– 11). MPICH-1.2.5 and MPICH-2 both use the same algorithm for broadcast, so the difference in performance is only due to the overall faster communication in MPICH-2.

2.3.3 Subtask: Parallel File System

Parallel file systems facilitate parallel access to data from parallel program. If applications are I/O intensive (clearly the case in the data warehousing, mining and analysis domain and subject of this proposal), and if I/O and disk accesses are done sequentially, the advantages of parallel computations will be lost. In high-performance computers, consisting of expensive disk arrays and specialized software, which is non-portable, expensive and works on specific vendor platforms, a portable parallel file system for cluster environments is highly attractive. As a part of this task, have ported the evaluate, adapt, port and install a parallel file system called "Parallel Virtual File System (PVFS)" available in the public domain to the cluster environment (and described in more details in the proposal). Performance evaluation is under progress.

Experiments

We installed the PVFS parallel file system on the cluster. PVFS is a popular open-source parallel file system for Linux clusters being developed at Argonne National Laboratory and Clemson University (`http://www.parl.clemson.edu/pvfs/`). PVFS allows multiple clients to concurrently write to or read from a single file. For this purpose, it uses multiple I/O servers and files are striped across the servers. The file striping is transparent to the user; the PVFS client software knows which parts of a file are stored on which servers. We installed PVFS on four nodes of our cluster. The PVFS I/O daemons (servers) were

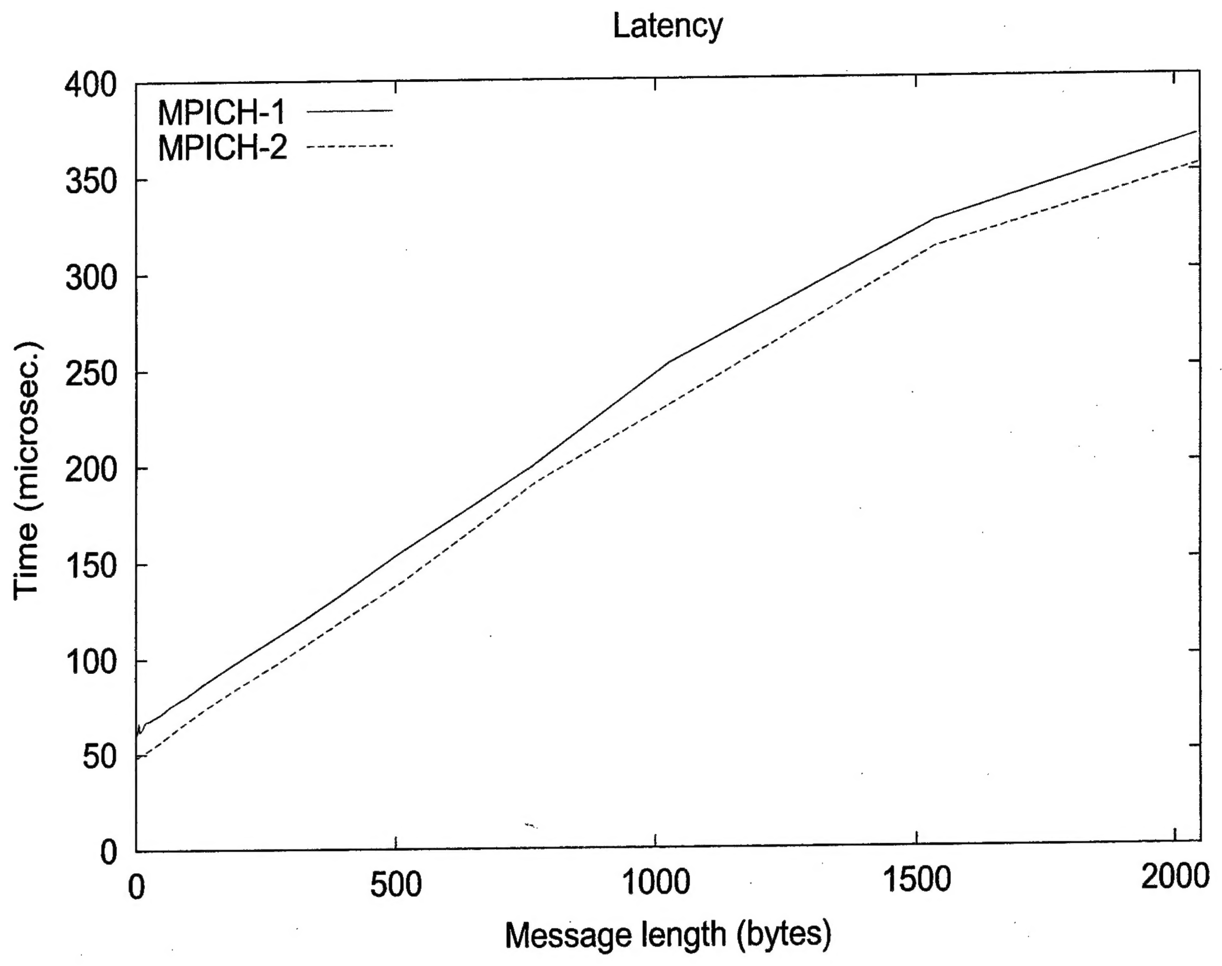

Figure 7: MPICH-1 versus MPICH-2 communication latency

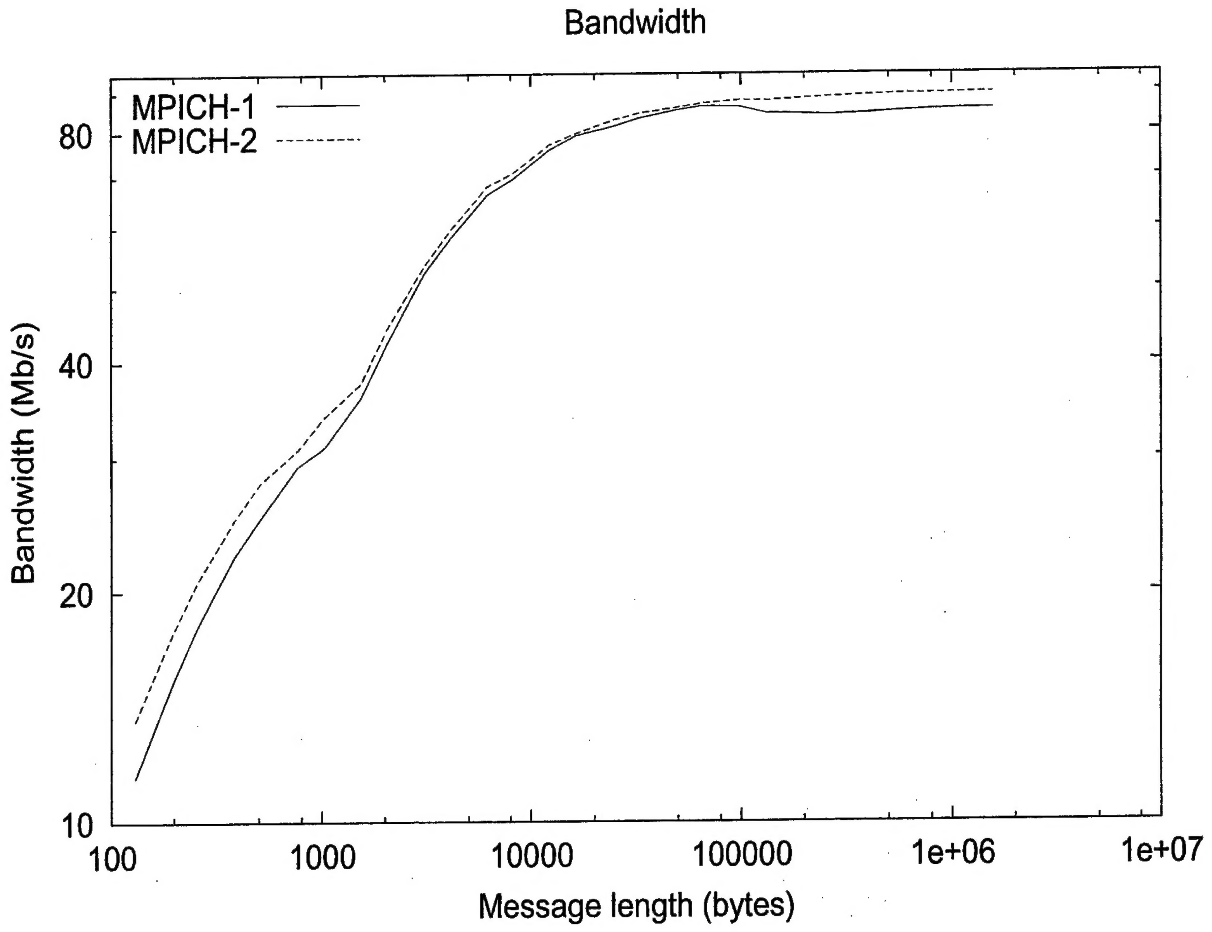

Figure 8: MPICH-1 versus MPICH-2 communication bandwidth

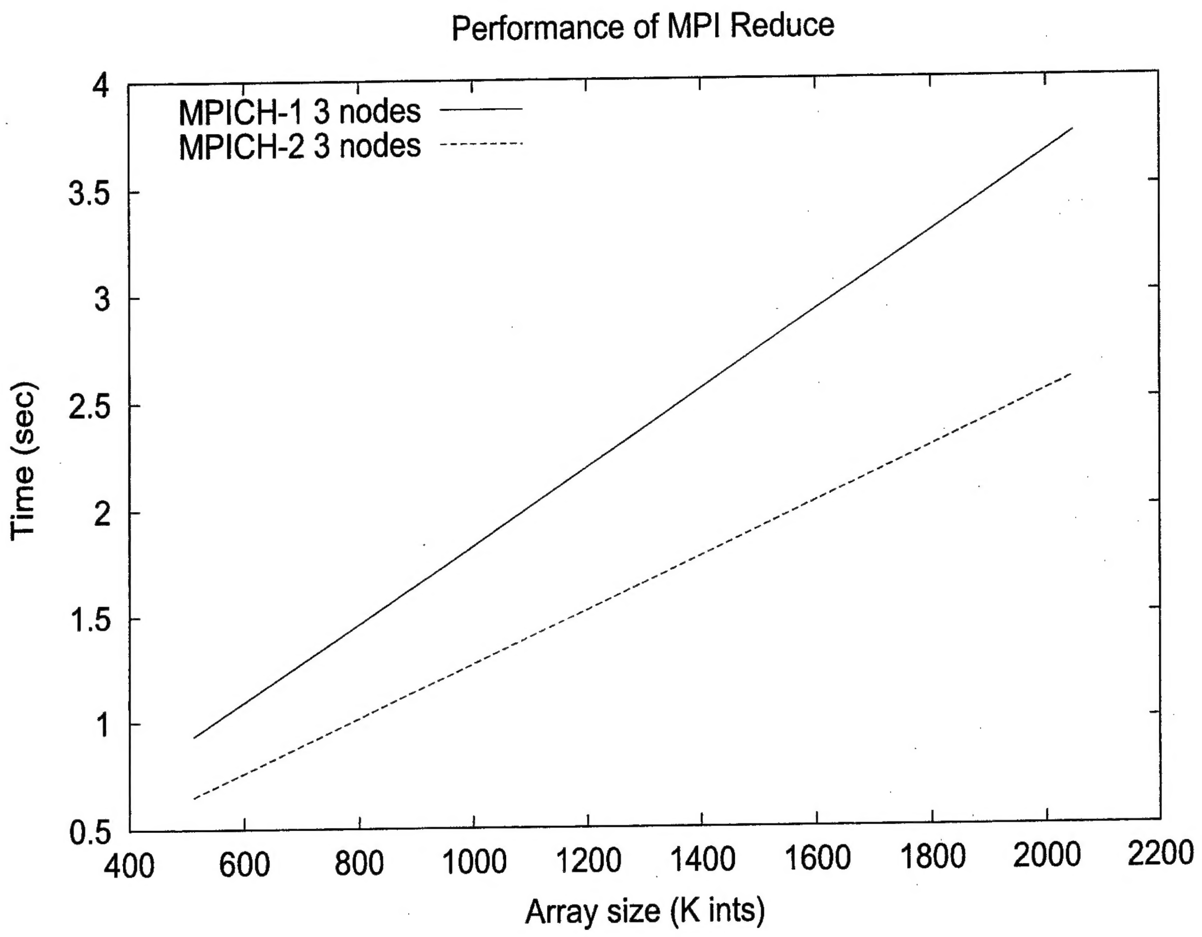

Figure 9: Performance of `MPI_Reduce` on 3 nodes

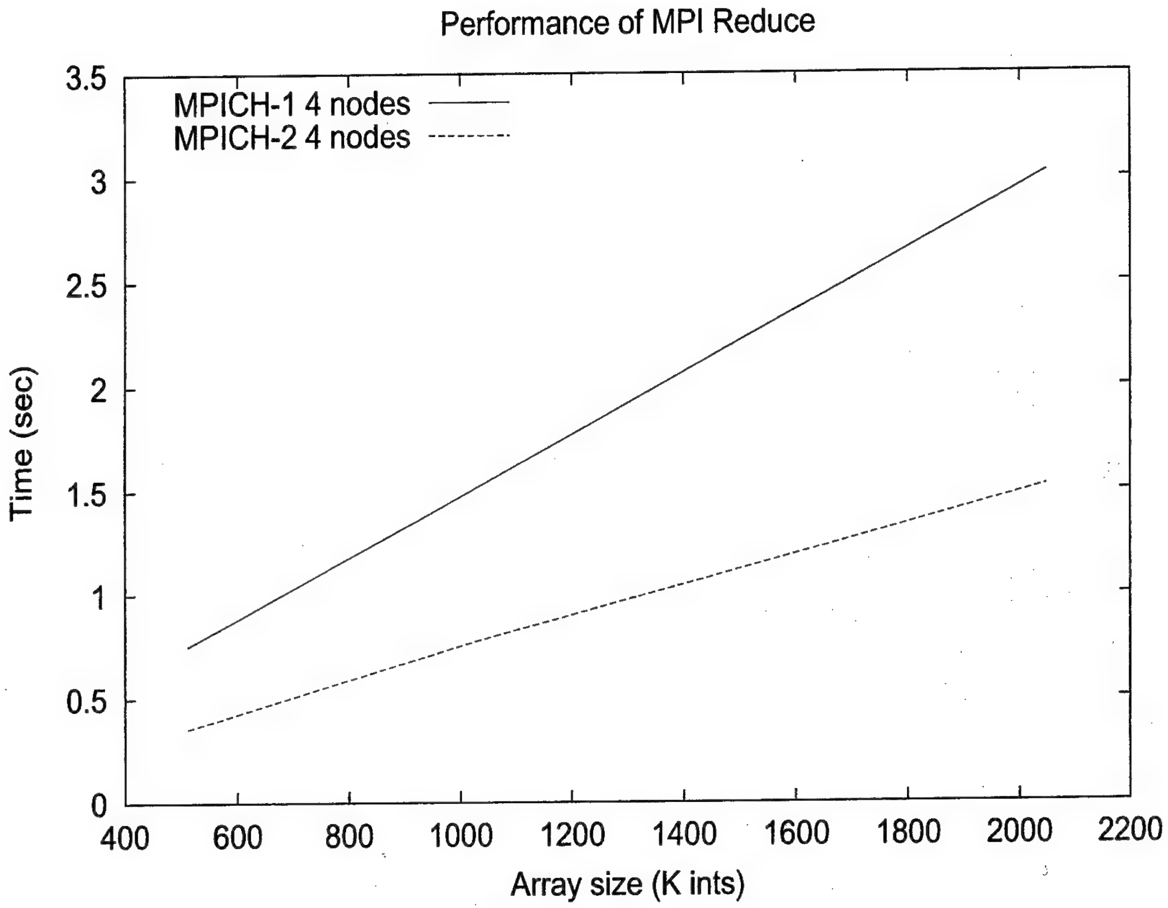

Figure 10: Performance of `MPI_Reduce` on 4 nodes

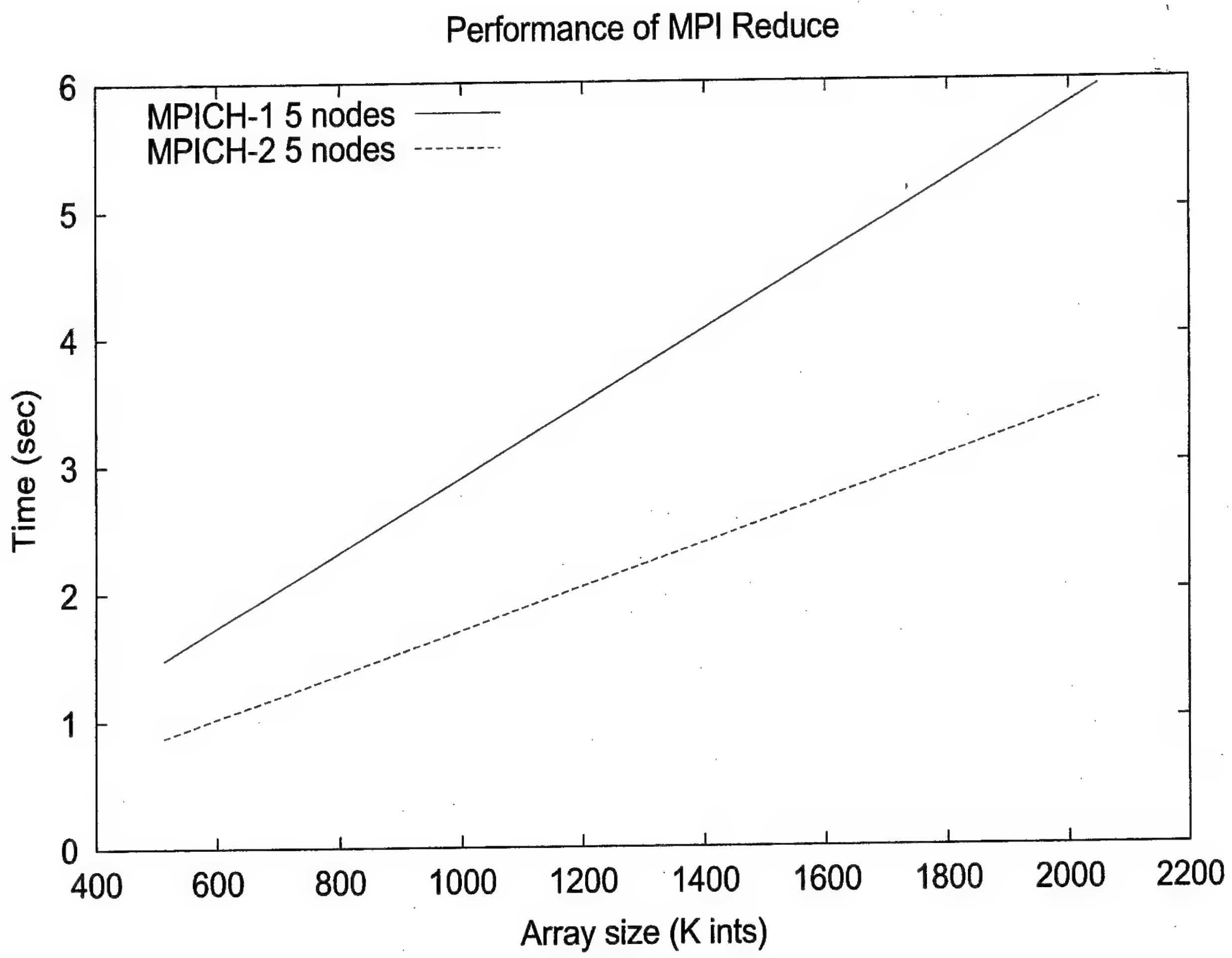

Figure 11: Performance of `MPI_Reduce` on 5 nodes

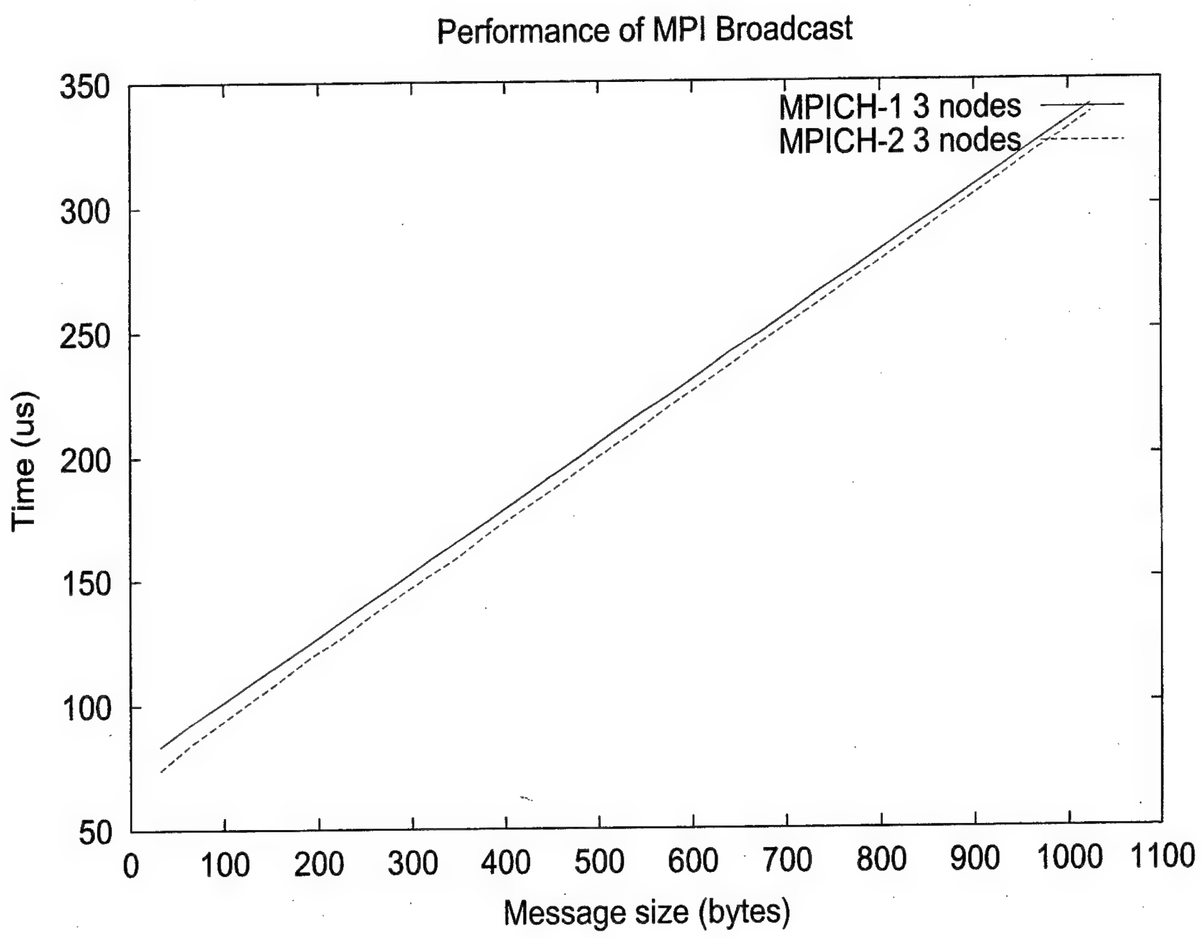

Figure 12: Performance of `MPI_Reduce` on 3 nodes

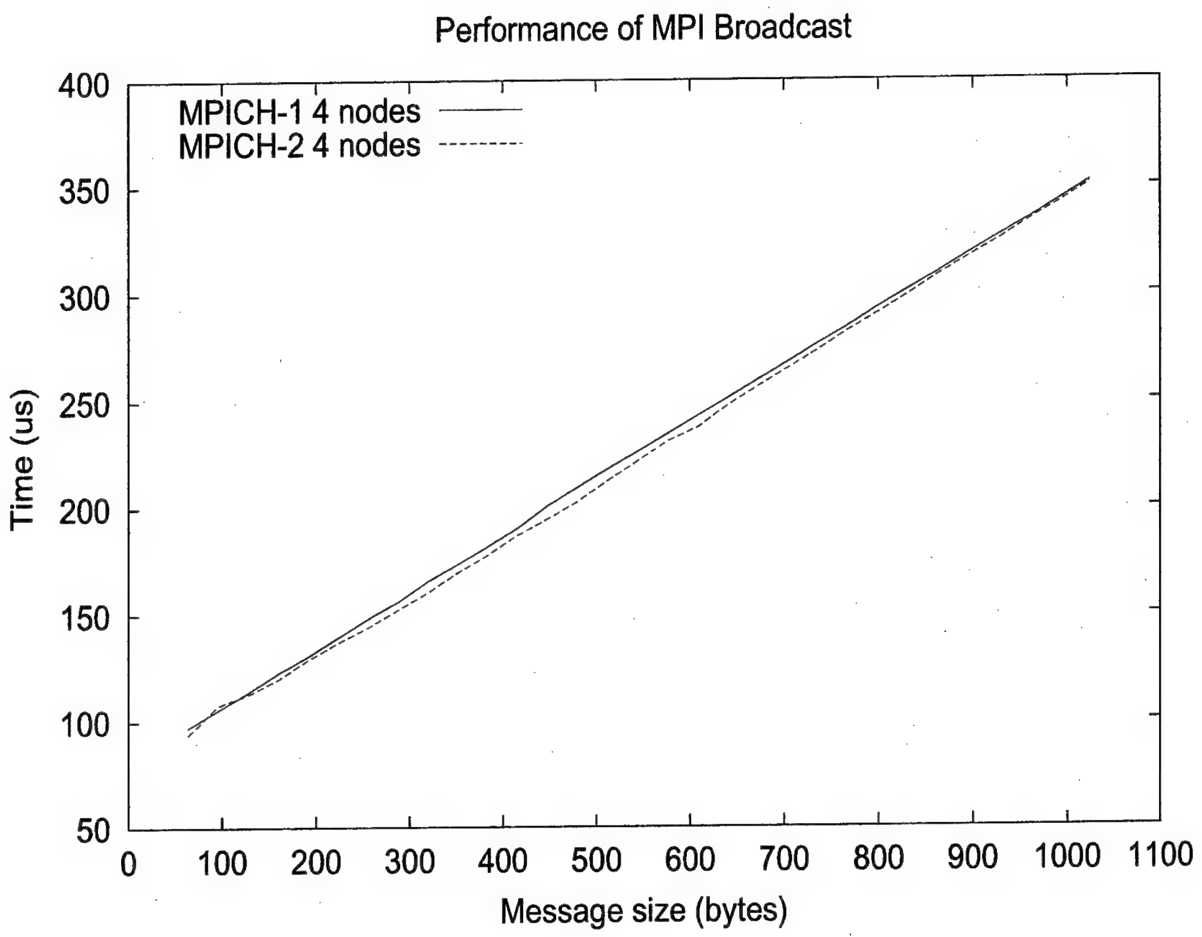

Figure 13: Performance of `MPI_broadcast` on 4 nodes

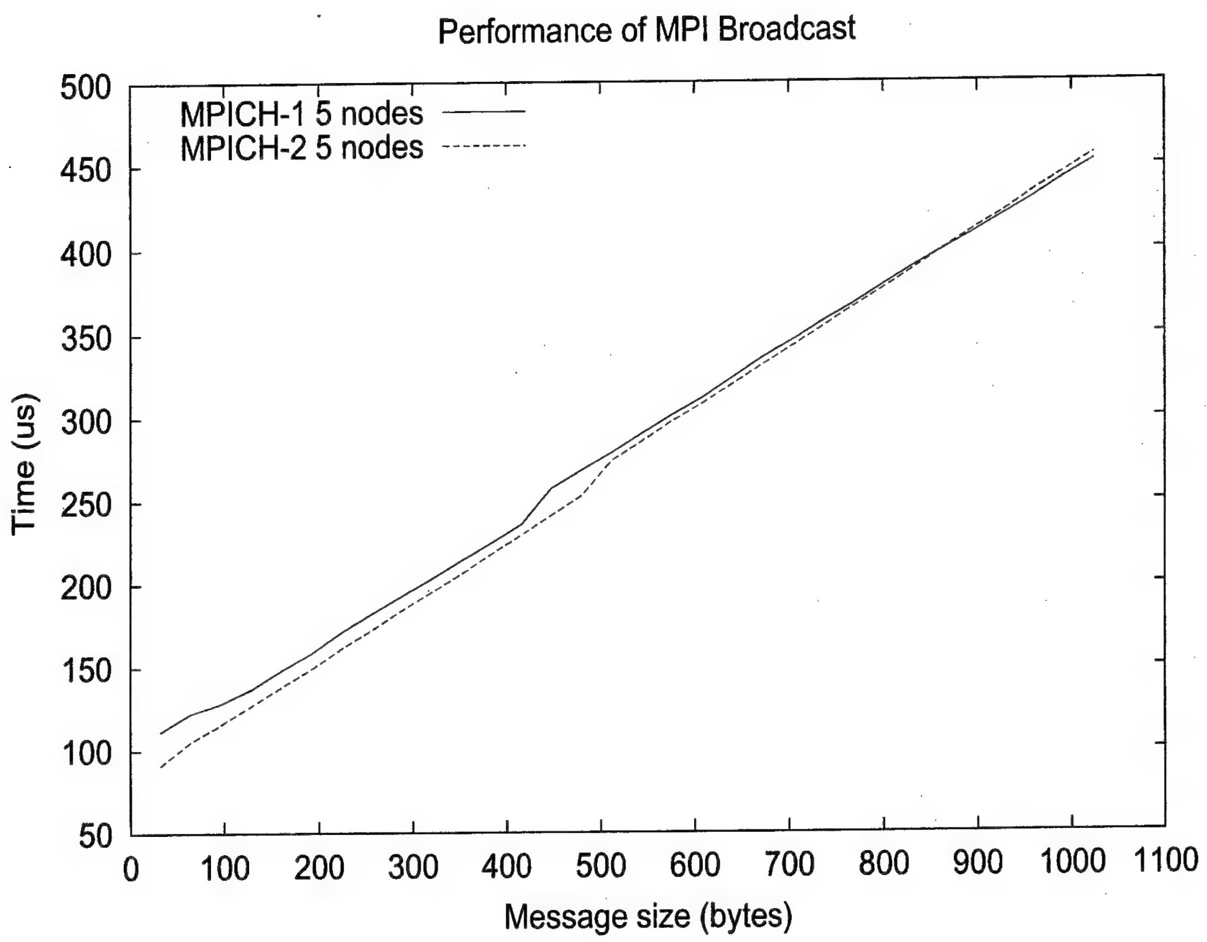

Figure 8: Performance of `MPI_Reduce` on 5 nodes

Figure 14: Performance of `MPI_broadcast` on 5 nodes

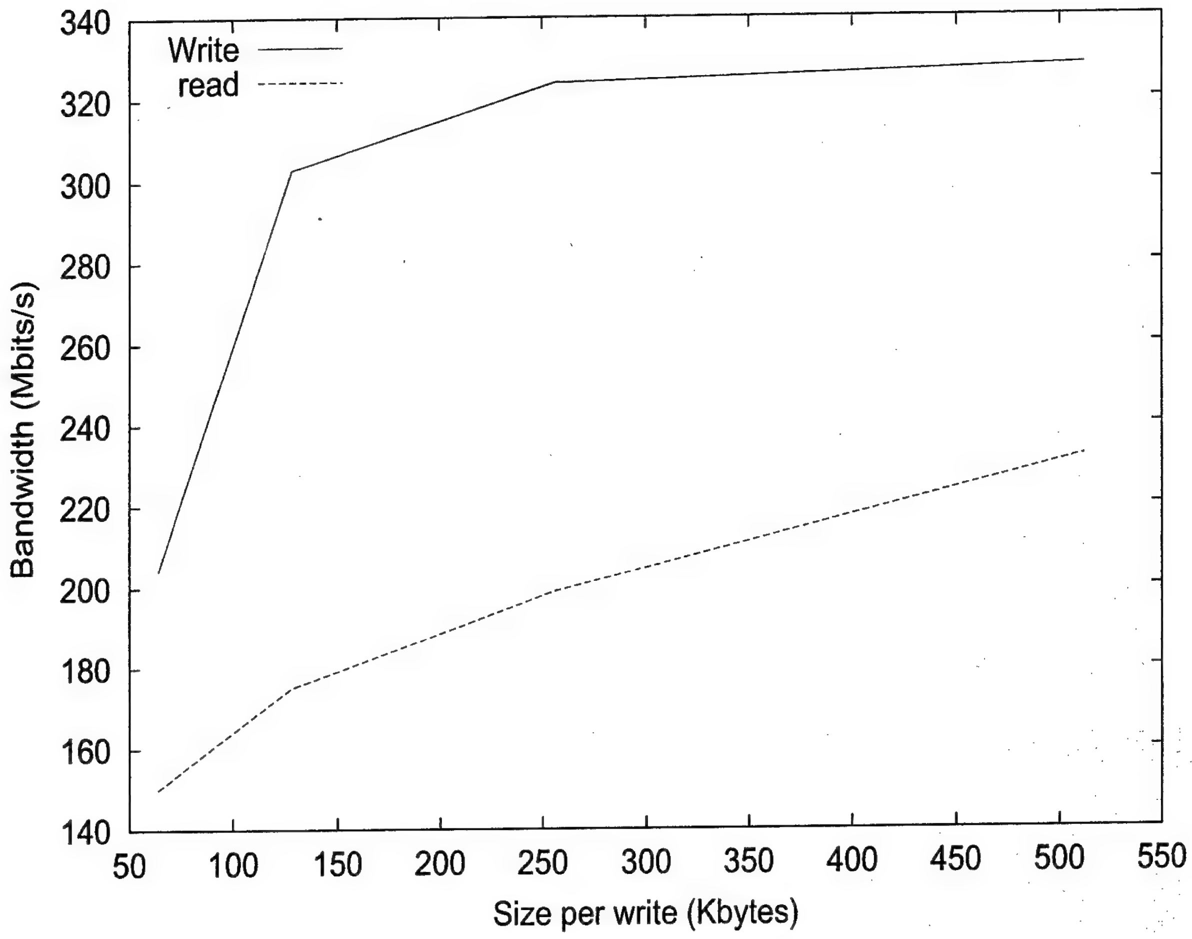

Figure 15: Parallel I/O bandwidth with PVFS

running on the four nodes, with each server using the local disk on the node to store the distributed file. We ran some basic tests to measure the bandwidth for concurrent reads and writes to a common file. We used a parallel MPI program running on all four nodes, in which each process opened a common file and wrote to (or read from) different parts of the file 100 times. We varied the size of each write and read. The results are shown in Figure 15. Note that the bandwidth is limited by the fast ethernet connection between the nodes and hence is limited to 400 Mbits/s on four nodes. The write bandwidth increases as we increase the size of the write and reaches up to 330 Mbits/s. The read bandwidth is lower, reaching up to 232 Mbits/s for the data sets chosen. The write bandwidth is higher because the writes get cached on the PVFS servers before being written to disk. In the case of reads, the data must be read from the disk where it resides.

2.3.4 Subtask: Node Database System

We incorporated the MySQL database system, which is the world's most popular open source database (http://www.mysql.com/). Its architecture lends itself to high performance and easy customization. Extensive reuse of code within the software and a minimalist approach to producing functionally-rich features has resulted in a database management system with high-performance, compactness, stability and ease of deployment. The unique separation of the core server from the storage engine makes it possible to run with strict transaction control or with ultra-fast transaction-free disk access, whichever is most appropriate for the situation. We have incorporated the MySQL database systems on the nodes of our cluster as proposed in the proposal.

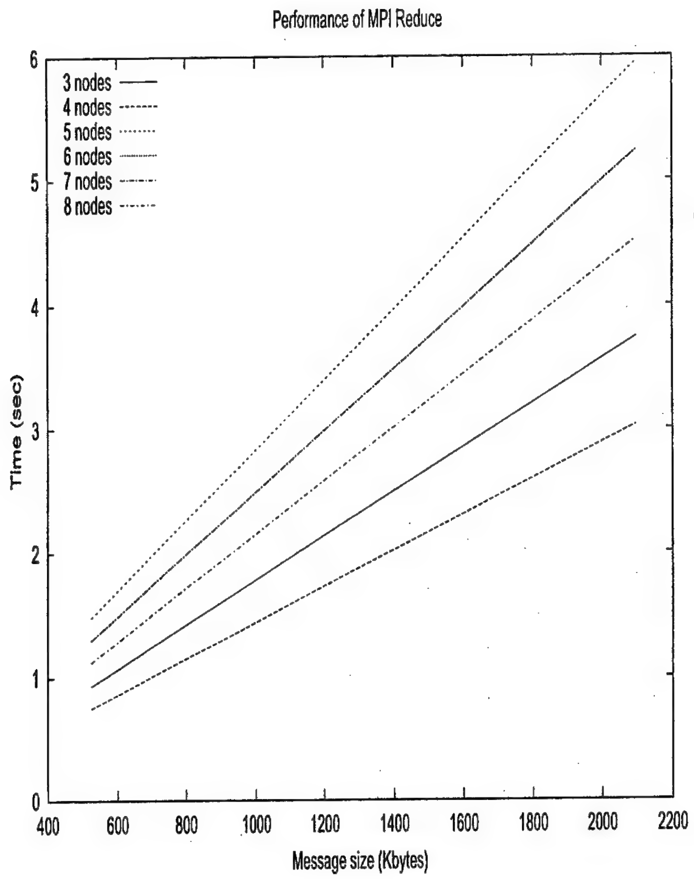

Figure 16: This figure illustrates the performance of the sub/reduce parallel primitive which is one of the most common operations found in most statistical and data mining algorithms. Clearly, good performance and scalability are obtained for this operation on our cluster.

2.4 Task: Parallel primitives and Functions

Aggregate and reduction operations are two important parallel primitives as they are needed in almost all statistical and data mining function, including in slicing data along one or more dimensions, computation of aggregate along one or more dimensions, slicing and dicing of data, accessing data in ranges along one or more dimensions, functions to count, compute statistics, frequent set calculations (such as required in the Association rules discovery algorithms), computations of distributions and histograms required for multidimensional clustering and others.

We parallelized, implemented and tested the performance of MPI Reduce for global sum operations for integers. The performance results are in Figure 16. It also shows the binary tree nature of the algorithm: power-of-two number of processes perform better than non-power-of-two number processes. Clearly, good performance and scalability are obtained for this operation on our cluster.

2.5 Task: Parallel Multidimensional Clustering

We proposed to design, develop, implement and evaluate a parallel version of a multidimensional clustering algorithm. Clustering is a very common mining technique to determine clusters of similar records in a populations.

Data clustering techniques help discover interesting patterns (called clusters) from large data sets. Such patterns often exist in high-dimensional space. Efficient data clustering techniques address data and noise that both exist in high-dimensional spaces and subspaces, which result in an exponential growth of the search space for clusters. We implemented a data clustering primitive based on the parallel adaptive grid and density-based algorithm called MAFIA [5]. Details of this algorithm are described in the proposal are not presented again.

The data-clustering primitive just discussed was implemented and executed over a data set of 4,281,782 records (> almost 0.5GB), which contains 20-dimensional data with 5 clusters and each cluster is of 5 dimensions. We experimented with the cluster for 2, 4, and 8 processors, using data page sizes of 8K, 16K, 32K and 64K bytes. Figure 17 presents the total response times of the clustering primitive with respect to increasing number of processors across all the page sizes (the amount of data accessed from the disk in one request). The results show a reduced response time as we increased the data page size, on account of better disk utilization. Clearly, scalable performance is achived for this algorithm.

2.6 Task: Parallel Association Rules

We proposed to design, develop, implement and evaluate a parallel version of the Association Rules data mining (ARM) algorithm on the cluster. Association rules are commonly used for finding patterns and rules of events that occur together, purchase patterns, and for prediction. The algorithm is described in detail in the proposal, and therefore, is not presented here again for the sake of brievity.

Typically, an ARM process consists of two major phases. During the first phase, a set of frequent itemsets is found. An itemset is frequent if its support is greater than a given threshold S. Then, in the second phase, the rules that satisfy the minimum S and the minimum confidence C are identified. The first phase, which performs the Frequent Set Counting (FSC), tends to be much more expensive than the second phase, especially with a low S, since I usually contains a large number of distinct items. To evaluate ARM FSC workload, we have adopted an FSC primitive based on the work proposed in [1]. The FSC primitive exploits the total memory of a cluster system for higher efficiency.

To evaluate the FSC primitive, a synthetic transactional database was generated using the data generator similar to that available from IBM, which is used to evaluate ARM algorithms [4]. The database had an average transaction size of 20, and each transaction can contain up to 25 distinct items. 100,000 transactions were generated for each processor, with a minimum support of 5%. Figure 18 presents our results for the FSC primitive executed on our cluster with 2, 4, and 8 processors. It should be noted that the data size is also scaled linearly with the number of processors. For example, data size used for 4 processors is twice

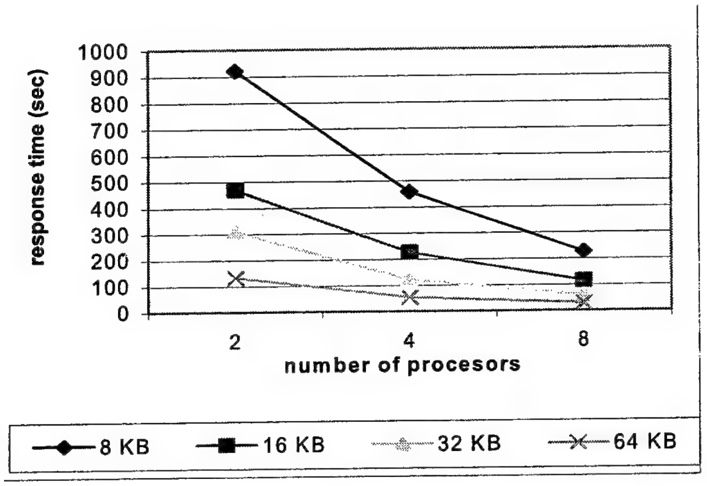

Figure 17: Implementation and Evaluation of the Parallel Multidimensional Clustering Algorithm on the Linux Cluster. Input data includes more than 4 million records. The graph shows almost linear scalability with the number of processors even though the network is only a 100Mb ethernet. It is also clear that use of different block (page) sizes from disks provide scalable performance even though higher block size results in lower access costs, and therefore, provides better performance.

that used for 2 processors. Therefore, for complete scalability, the response time should remain constant or increase slightly.

The results indicate that the FSC primitive provides scalable performance with increasing data size. Again, that the data size increases linearly with the number of processors, and therefore, keeping the response time constant demonstrates scalability with data size as well as with number of processors in the cluster.

3 Milestone/Task Status

3.1 Overview

The remaining subsections describe progress on the major tasks. For each task, we describe the following:

Schedule Whether the task is on schedule. We use the term **in progress** to describe a task that is partially or nearly completed and **not started** to describe a task that has not been started yet.

% Completion An estimate of the percentage completed for any task **in progress**.

Testing Program Not applicable to this project.

Designs Completed Summary of designs and implementations. In the context of software-related projects, we interpret designs as prototypes developed thus far and any supporting design/architectural sketches.

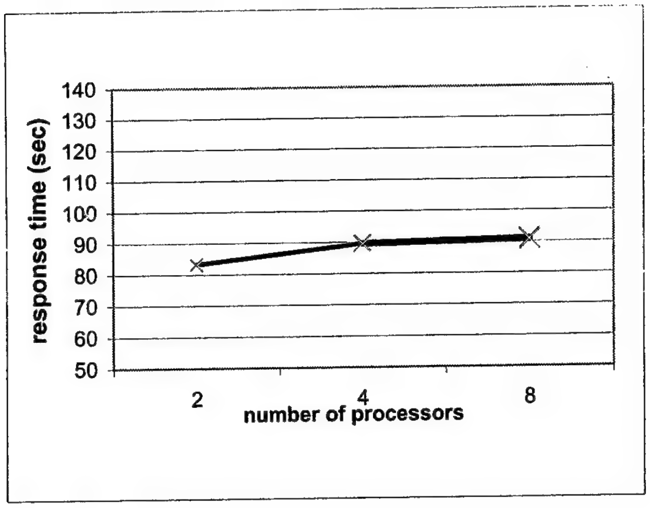

Figure 18: Implementation and Evaluation of the Parallel Associations Rules Discovery Algorithm on the Linux Cluster.The database had an average transaction size of 20, and each transaction can contain up to 25 distinct items. 100,000 transactions were generated for each processor, with a minimum support of 5%. It should be noted that the data size is also scaled linearly with the number of processors.

3.2 Task: Scalable Commodity-based Cluster

Schedule

completed

% Completion

100%

Testing Program

Not applicable.

Designs Completed

As illustrated earlier in the report, we have designed, developed and built the terabyte cluster. Figure 1 shows the figure of the cluster, along with the RAID storage system. Specifications of the cluster are described earlier.

3.3 Task: Software Architecture

Schedule

completed

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We have finalized the design of the software architecture as proposed in the proposal. Figure 2 shows the architecture. If refinement is needed in the future due to required extensions, those refinements will be incorporated.

3.4 Task: Parallel Software Infrastructure

Schedule

Completed

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We have completed the communication software implementation and evaluation and its performance and scalability results are presented earlier. We have performed extensive evaluation and shown scalability on our cluster. Furthermore, we implemented two versions of the communication software and compared their performance. The parallel file system has been ported. We have also evaluated its performance. Performance results have been presented earlier. Node database system, MySQL has also been incorporated into the cluster and tested.

3.5 Task: Parallel Primitives

Schedule

Completed

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We have completed and evaluated parallel primitives, in particular the reduce and aggregation operations.

3.6 Task: Parallel Multidimensional Clustering

Schedule

completed

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We have implemented and evaluated the parallel multidimensional clustering algorithm based on multiple adaptive finite interval analysis. We evaluated the quality and scalability on our cluster infrastructure.

3.7 Task: Parallel Association Rules

Schedule

In progress

% Completion

100%

Testing Program

Not applicable.

Designs Completed

We have implemented and evaluated the parallel association rules algorithm based on hybrid data distribution to achieve effective load balancing and scalability. In our results we demonstrated scalability with data size as well as number of processors. Furthermore, we evaluated the algorithm on realistic data sets reflecting real-world transactions in different domains.

4 Outstanding Issues from Previous Report

None.

5 New Problems

None.

6 Conferences and Trips

Drs. Alok Choudhary and George Thiruvathukal visited Dr. Lt. Col. Doug Dyer at DARPA on July 23, 2003 to update him on the progress of our project as well as to obtain feedback.

7 Any Potential Impacts on Schedule

None.

8 Conclusions and Future Plans

We have completed all tasks and have been able to demonstratre the ideas in building commodity cluster, designing and evaluting software architecture as well as designing and implementing primitives and parallel data analysis algorithms. In fact, we were able to alternative implementations and evaluation for some cases. We believe, we have a sclable cost-effective cluster design and sclable software infrastructure to enable large data analysis algorithm. We would be interested in exploring larger projects based on this initial system.

9 Itemized Person-Hours and Costs

Description	*Quantity*	*Rate*	*Sub-Total*
PI (Choudhary)	90	$125	$11,250
Senior Personnel (Thiruvathukal) Labor	45	$125	$5,625
Student Interns	1000	$20	$20,000
Consultant (Thakur)	20	$100	$1,000

During this progress period, the PI, senior personnel, and student interns completed the aforementioned tasks.

References

1 E-H Han, G. Karypis and V. Kumar, "Scalable Parallel Data Mining for Association Rules". IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No. 3. May/June 2000.

2 S. Orlando, P. Palmerini, R. Perego and F. Silvestri, "An Efficient Parallel and Distributed Algorithm for Counting Frequent Sets". VECPAR 2002 High Performance Computing for Computational Science. June 2002.

3 R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules". IEEE Transactions on Knowledge and Data Engineering, 8(6): 962-969. December 1996.

4 IBM Quest Data Mining Project. Quest website: http://www.almaden.ibm.com/cs/quest/syndata.html.

5 H. Nagesh, S. Goil and A. Choudhary, "Parallel MAFIA: Parallel Subspace Clustering for Massive Data Sets". Data Mining for Scientific and Engineering Applications. Academic Publishers. 2001.